Area-throughput trade-offs for SHA-1 and SHA-256 hash functions' pipelined designs

MICHAIL, Harris E., ATHANASIOU, George S., KELEFOURAS, Vasileios <http://orcid.org/0000-0001-9591-913X>, THEODORIDIS, George, STOURAITIS, Thanos and GOUTIS, Costas E.

Published version

MICHAIL, Harris E., ATHANASIOU, George S., KELEFOURAS, Vasileios, THEODORIDIS, George, STOURAITIS, Thanos and GOUTIS, Costas E. (2016). Area-throughput trade-offs for SHA-1 and SHA-256 hash functions' pipelined designs. Journal of Circuits, Systems and Computers, 25 (04), p. 1650032.

Journal of Circuits, Systems, and Computers

AREA-THROUGHPUT TRADE-OFFS FOR SHA-1 AND SHA-256 HASH FUNCTIONS' PIPELINED DESIGNS

--Manuscript Draft--

Manuscript Number:	WSPC-JCSC-D-15-00085R2
Full Title:	AREA-THROUGHPUT TRADE-OFFS FOR SHA-1 AND SHA-256 HASH FUNCTIONS' PIPELINED DESIGNS
Article Type:	Research Paper
Keywords:	Hash function; Message Authentication Code; Pipeline; FPGA; Security.
Corresponding Author:	HARRIS E MICHAIL, Dr. CYPRUS
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	
Corresponding Author's Secondary Institution:	
First Author:	HARRIS E MICHAIL, Dr.
First Author Secondary Information:	
Order of Authors:	HARRIS E MICHAIL, Dr.
	George Athanasiou
	Vasileios Kelefouras
	George Theodoridis
	Thanos Stouraitis
	Costas Goutis
Order of Authors Secondary Information:	
Abstract:	High-throughput designs of hash functions are strongly demanded due to the need for security in every transmitted packet of worldwide e-transactions. Thus, optimized and non-optimized pipelined architectures have been proposed arising, however, important questions. Which is the optimum number of the pipeline stages? Is it worth to develop optimized designs or the same results can be achieved by increasing only the pipeline stages of the non-optimized designs? The paper answers the above questions studying extensively many pipelined architectures of SHA-1 and SHA-256 hashes, implemented in FPGAs, in terms of throughput/area factor. Also, guides for developing efficient security schemes designs are provided.
Response to Reviewers:	

Journal of Circuits, Systems, and Computers

AREA-THROUGHPUT TRADE-OFFS FOR SHA-1 AND SHA-256 HASH FUNCTIONS' PIPELINED DESIGNS

HARRIS E. MICHAIL†

Electrical Engineering and Information Technology Department, Cyprus University of Technology
Lemesos 3036, Cyprus
†*harris.michail@cut.ac.cy*

GEORGE S. ATHANASIOU, VASILEIOS I. KELEFOURAS, GEORGE THEODORIDIS, THANOS STOURAITIS, COSTAS E. GOUTIS

VLSI Design Laboratory, Electrical and Computer Engineering Department, University of Patras
Patras, GR-26504, Greece

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

High-throughput designs of hash functions are strongly demanded due to the need for security in every transmitted packet of worldwide e-transactions. Thus, optimized and non-optimized pipelined architectures have been proposed arising, however, important questions. Which is the optimum number of the pipeline stages? Is it worth to develop optimized designs or the same results can be achieved by increasing only the pipeline stages of the non-optimized designs? The paper answers the above questions studying extensively many pipelined architectures of SHA-1 and SHA-256 hashes, implemented in FPGAs, in terms of throughput/area factor. Also, guides for developing efficient security schemes designs are provided.

Keywords: Hash function, Message Authentication Code, Pipeline, FPGA, Security

1. Introduction

High-throughput designs of security schemes are highly needed nowadays, since security services have become an inseparable feature of almost all e-transactions. A crucial module of these schemes is authentication, which is performed using a cryptographic hash function. Hash functions are widely used as sole cryptographic modules or incorporated in hash-based authentication mechanisms like the Hashed Message Authentication Code (HMAC).[1]

Applications that usually employ hash functions include the Public Key Infrastructure (PKI),[2] Secure Electronic Transactions (SET),[3] the IEEE 802.16 standard for Local and Metropolitan Area Networks,[4] and digital signature algorithms, like DSA,[5] which are used to provide authenticating services in applications such as elec-

tronic mail and data interchange, electronic funds transfer, software distribution, etc. Hash functions are also used in Secure Sockets Layer (SSL)[6] and Transport Layer Security (TLS),[6] which are protocols for establishing authenticated and encrypted sessions between servers and clients. Also, hash functions are required to provide authentication services to Virtual Private Networks (VPNs).[7]

Lately, further attention has been drawn to hash functions because of their usage in the Internet Protocol Security (IPSec).[7] IPSec is a mandatory feature in the forthcoming Internet Protocol version 6 (IPv6),[8] which will be massively adopted.[9] In order to provide a high of security, IPSec incorporates encryption and authentication schemes. Encryption is provided through the block cipher algorithm AES,[10] whereas authentication is offered through HMAC that is built on top of a standard cryptographic hash function (e.g., MD-5 or SHA-1).[11]

While security problems have been discovered in SHA-1[12] and MD-5[13] hash functions, the problems of SHA-1 are considered as non-critical. Thus, except SHA-1, the SHA-2 hash family is expected to be adopted as a secure solution in security schemes (e.g., IPSec/IPv6) in the forthcoming years.[14] Also, the US National Institute of Standards and Technology (NIST), has established a competition for developing the new hash function standard (SHA-3), to be finalized by the end of 2012.[15] As the transition to a new standard does not happen immediately, SHA-1 and SHA-2 families are expected to continue being used in near- and medium-future applications.[14]

IPv6 (and other applications that need cryptographic processing in every transmitted data packet) have to be able to operate at high transmission rates, for instance over 30 Gbps which is currently offered by optical networks.[16] This is prohibited by the included security schemes[17] that usually incorporate a hash function and a block cipher algorithm. Hence, hardware implementations are needed in order to achieve high-throughput processing. Many designs for the AES algorithm implemented in FPGA or ASIC technologies have been proposed achieving high throughput values ranging from 20 up to 70 Gbps.[16,18] The same holds for SHA-1 and SHA-2 functions where design methodologies and optimized implementations on FPGAs have been introduced achieving throughputs up to 11Gbps.[19,20,21,22]

In order to improve delay and throughput, pipeline-based designs have been used extensively. Specifically, in the vast majority of the hash functions implementations, a four-stage pipeline architecture is adopted.[19,20,21,22,23,24,25,26,27,28,29] The most possible reason for this decision is that the SHA-1 function uses four different non-linear functions for the operations that constitute the main hash calculations (cf. Section 3). In these designs, each non-linear function is used for a certain number of the iterations of the algorithm. This results in a straightforward solution that uses four pipeline stages, avoiding extra circuitry and complex control logic while keeping the area budget low. This design choice was also adopted for the design of SHA-2 family, although the non-linear functions are the same for all iterations of the algorithms.

However, no study has ever been performed on the optimum number of pipeline

stages in the hash function designs, in terms of its impact on major design metrics such as delay, throughput, and throughput/area ratio. Therefore, there is no concrete justification that the 4-stage pipeline architecture is the optimal choice for achieving the best implementation for the SHA-1 hash function or SHA-2 hash family.

Also, as complex design methodologies and techniques have been proposed,[19,20] to develop high-throughput optimized designs for the aforementioned hash functions, an additional question arises, whether all these methods are worth their complexity. In other words, could we achieve the same or even better results, in terms of delay, throughput, and throughput/area ratio by simply increasing the number of pipeline stages of the non-optimized designs? If yes, what would be the impact on the area? Moreover, in the case of the optimized designs, what is the optimal number of pipeline stages?

The paper answers all the above questions by performing a comparative analysis of several pipelined versions of the SHA-1 and SHA-256 hash functions implemented on various Xilinx FPGA families and studying the trade-offs in terms of performance (delay), throughput, and throughput/area factors. This analysis is performed for base (non-optimized) designs, which are implementations derived straightforward as implied by the standard[11] and also for the optimized designs of SHA-1 and SHA-256 hash functions, as proposed in.[19,20] It should be noticed that the SHA-256 function was selected from the SHA-2 family as it is the most widely used function of this family. The above analysis can be very useful during the development of complex security schemes that include a hash function as a major module, since it specifies the bounds of the hash functions designs in terms of throughput and throughput/area factors in the considered FPGA families. For that reason, in companion with existing studies for the AES algorithm,[16,30,31] design guides are also provided towards the development of high-throughput designs for advanced security schemes like IPSec.

The rest of the paper is organized as follows: Section 2 presents the related work, while Section 3 provides the SHA-1 and SHA-256 hash functions background. Section 4 describes the SHA-1 and SHA-256 base and optimized architectures. Section 5 provides the experimental results performed on a large number of pipelined designs of the above hash function architectures implemented on a series of Xilinx FPGA families, and in Section 6 the performance trade-offs investigation for the base and optimized architectures of SHA-1 and SHA-256 hash functions are explicitly discussed. In Section 7, the design guides suggested by the trade-offs' investigation along with a general discussion are presented. Finally, Section 8 concludes the paper.

2. Related Work

SHA-1 and SHA-2 hash standards were announced by NIST in 2002.[11] Since then a variety of hardware implementations has been proposed. Existing studies ini-

tially proposed hardware implementations of SHA-1 and SHA-256 functions without paying much effort to throughput.[32,33] There are also designs that do not apply pipeline,[21,34,35,36,37,38] as they were mainly targeting area or focusing on the optimization of the transformation round only.

Later on, more complex implementations were proposed applying advanced design techniques and methods such as pipeline, hardware reuse, and parallelism exploitation.[23,24,25,26,28,29] However, the growing need for high throughput cryptographic designs led to studies proposing more sophisticated methods to optimize hardware implementations of the hash mechanism, such as algorithmic- and system-level optimization techniques (e.g., retiming, pre-computation, resource rescheduling, and loop-unrolling) which aim to optimize both the transformation round (as explained in Section 3) of the hash function and the whole system design.[19,20,21,22]

There are also many commercial Intellectual Property (IPs) designs implementing the SHA-1 and SHA-256 cores that incorporate certain optimization techniques, which are not revealed for commercial reasons.[39,40,41] However their performance lags that of designs offered by.[19,20]

All previously published works on SHA-1 and SHA-2 implementations that employed pipelining incorporate a pipeline of four stages; however, none of these works justifies the selection. Instead, the authors simply adopt a four-stage pipeline architecture by using four operational rounds, separated by pipeline registers as guided by the logic variation of SHA-1 algorithm. The application of 4-stage pipeline, when targeting high-throughput hardware designs, was also adopted for SHA-2 hash family algorithms without any motivation about this decision.

It has to be stressed that, no previous work has been presented that explores the number of the employed pipeline stages in SHA-1 and SHA-2 hash functions' designs as a design optimization parameter that affects performance, throughput, and throughput/area trade offs. A similar work exists for the AES block cipher algorithm. Specifically, the throughput/area trade-offs were explored for a wide range of pipelined implementations for ASIC[16] and FPGA technologies.[30,31] Thus, in conjunction with these works, the proposed one can be used as a designers guide for developing optimized designs in terms of performance, throughput,

and area of widely used security schemes and especially for the whole IPSec, used in IPv6.

3. SHA-1 and SHA-256 Hash Function Background

A hash function, $H(M)$, operates on an arbitrary-length message, M, and returns a fixed-length output, h, which is called the *hash value* or *message digest* of M. The aim of hash function is to provide a "signature" of M that is unique. Given M it is easy to compute h if $H(M)$ is known. However, given h, it is hard to compute an M such that $H(M) = h$, even when $H(M)$ is known.

Hash functions are iterative algorithms, which perform a number of iterations

called *transformation rounds* or *operations*, which include identical or slightly varying arithmetic and logical computations. As shown in Figure 1, the hash function's computation consists of two main stages, namely the *preprocessing* and *computation*.[11]

In the preprocessing stage, the input message, M, is padded to ensure that the size of the padded message is a multiple of 512 or 1024 bits (according to the employed algorithm). Then, the padded message is parsed into k-bit blocks and the initial hash values, $H(0)$, are set. In the computation stage, a message schedule is performed on the k-bit blocks producing the W_t values, which each of them is fed to the corresponding t-th iteration of the transformation round. The transformation round takes as input the corresponding W_t value, a constant value K_t defined by the standard, and the $H(0)$ values (in the first iteration) or the values generated during the previous iteration and then performs word-level processing, which includes additions, non-linear function, and logical computations. Finally, after a certain number of iterations, the hash value or message digest is generated.

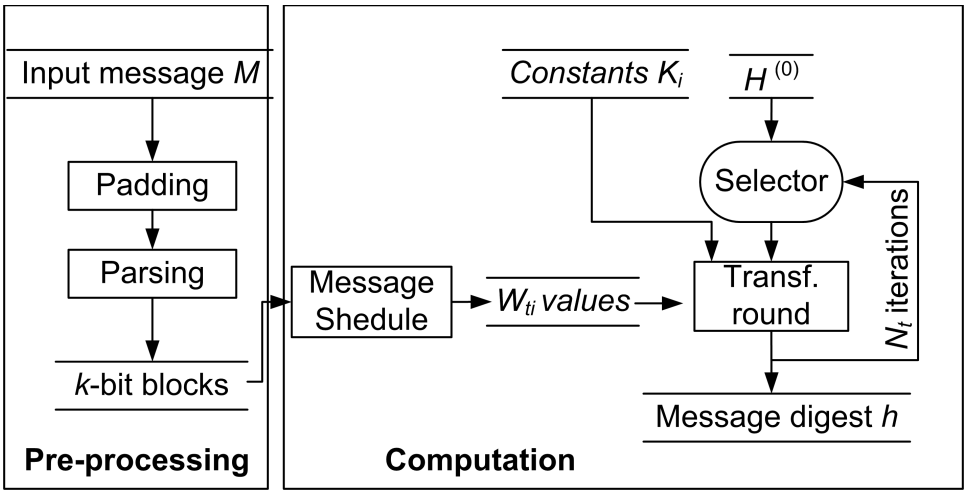

Fig. 1. Hash function computation

Concerning the SHA-1 function, the sizes of the k-bit block, word, and message digest equals to 512, 32, and 160 bits, respectively, whereas 80 iterations of the transformation round are required. Regarding the SHA-2 hash family, the most widely-used algorithm is the SHA-256 hash function for which the sizes of the k-bit block, word, and message digest equals to 512, 32, and 256 bits, respectively, whereas it requires 64 iterations. For more details the reader is referred to.[11]

4. SHA-1 and SHA-256 Base Architectures

In this section, the base and the optimized architectures for the SHA-1 and SHA-256 functions, which are used next in the comparative study, are presented. Specifically, we present the implementation of the transformation round for each function and the whole organization of each architecture. For the base versions, the transformation round is implemented exactly as described by the standard without performing any optimization. For the optimized versions, the architectures of [19,20] are used since they are among the best published designs in terms of throughput and throughput/area.

4.1. *Base architectures*

The base transformation round of SHA-1 hash function is depicted in Figure 2. During the t-th iteration ($t = 1, 2, \ldots, 80$), it receives five $(a_{t-1}, \ldots, e_{t-1})$ 32-bit words, performs the computations shown in Figure 2, and produces the output values $(a_t e_t)$. It should be noticed that during the first iteration the initial values, which are provided by the standard, are used. Also, the transformation round takes as input the W_{t-1} value produced by the message schedule unit and the K_{t-1} constant value also provided by the standard.[11]

The computations include modulo 2^{32} additions, rotations, $ROTL^n$, which correspond to n-times circular left rotation, and the function $f_t(x, y, z)$, which comprises 4 non-linear functions, which are given bellow:

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\bar{x} \wedge y) & 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z) = x \oplus y \oplus z & 60 \le t \le 79 \end{cases} \quad (1)$$

where $\wedge$ and $\oplus$ stand for logical AND and XOR operations respectively.

The base implementation of SHA-256 transformation round is shown in . It receives as inputs eight 32-bit words, $(a_{t-1}, \ldots, h_{t-1})$, the value W_{t-1}, and the constant value K_{t-1}, then performs the computations shown in Figure 3, and produces the values $(a_t, \ldots, h_t)$ after 64 iterations. Similar to SHA-1, the initial values are used in the first iteration. The incorporated computations include modulo 2^{32} additions and four non-linear functions of which $Ch(e, f, g)$ and $Maj(a, b, c)$ are the same as those of SHA-1 function (provided in Eq. (1)), whereas the two new ones $\sigma_0^{256}(x)$ and $\sigma_1^{256}(x)$ are given in Eq. (2).

$$\begin{aligned} \sum_{0}^{256}(x) &= ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \\ \sum_{1}^{256}(x) &= ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \end{aligned} \quad (2)$$

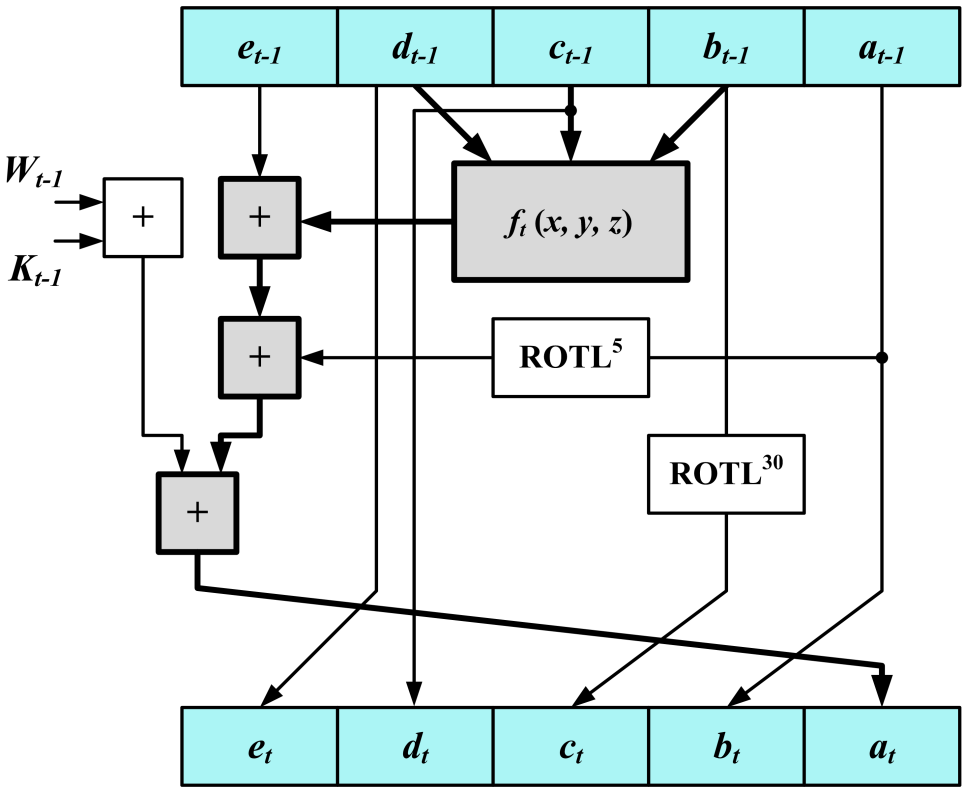

Fig. 2. SHA-1 hash function Base transformation round

where $ROTR^n$ means n-times circular right rotation.

The general pipeline architecture for the base SHA-1 and SHA-256 hash functions is shown in Figure 4. It consists of n pipeline stages each one including a round unit i ($i = 1, 2, \ldots, n$), which corresponds to the transformation round (Figure 2 and Figure 4), a W unit for producing the W_t values, and a constant memory, K, (organized as registers) for storing the constant values. Also, pipeline registers exist at the output of each round unit. For SHA-1, the W unit comprises XOR trees, whereas for SHA-256 it is realized by adders, rotators, and gates, as implied by the standard.

When the number of pipeline stages is smaller than the number of the algorithm's iterations, each stage executes more than one iteration. Thus, multiplexers are used in front of each stage to feed back the outputs of the current stage or to receive the output of the previous one. Also, m 32-bit adders ($m = 5$ for SHA-1 and $m = 8$ for SHA-256) are used to add the result of the n-th pipeline stage with the initial values as implied by the standard.

The control logic includes a set of counters each of which is used for addressing

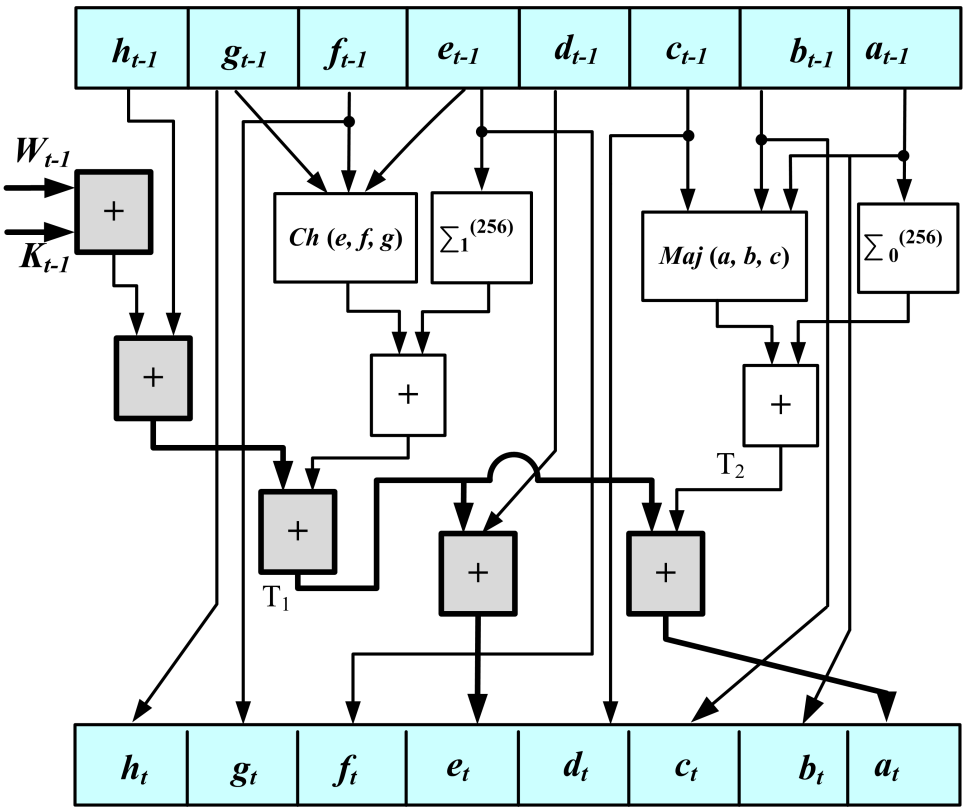

Fig. 3. SHA-256 hash function Base transformation round

the corresponding constant memory, controlling the multiplexer in front of the next stage round unit, and activating the counter of the next stage. Depending on the pipeline version and the implemented hash function, a counter in pipeline stage i counts up to the value required for each round unit to complete its computation, while the counter's output is used for addressing the corresponding constant memory. After the computation's completion of stage i, the counter is deactivated and the $tcround_i$ and $tcccount_i$ signals are generated to trigger the next pipeline stage.

Analyzing the architecture of Figure 4, the critical path is located in the round unit. The critical paths of the base rounds are shown with darker components in Figure 2 and Figure 3. The critical path of SHA-1 consists of three adders, a non-linear function and a multiplexer, whereas the SHA-256 one consists of four adders and a multiplexer. The multiplexers correspond to those in front of each round unit (Figure 4). These critical paths have also been verified by measurements performed on the implementations of the architecture.

As discussed in the introduction, all existing pipeline designs adopt a 4-stage architecture, with the hash value of the first message block being produced after

it has been processed in each pipeline stage for 20 iterations (spending 20 clock cycles) resulting in a total of 80 iterations, as long as SHA-1 is concerned. Due to the pipelined design, the hash value of each subsequent message block is produced after 20 cycles; thus, compared to the non-pipelined design, throughput is quadrupled. The same holds for SHA-256 with four pipeline stages. Each stage operates 16 times on the message block and after 64 clock cycles (64 iterations) the hash value is produced. An initial latency of 64 clock cycles is spent to produce the first message block and after that a new hash value is produced after 16 clock cycles for each subsequent message block.

4.2. *Optimized Architectures*

In [19,20] advanced optimization methodologies led to optimized SHA-1 and SHA-256 designs, which, when compared to the base ones, improve the throughput/area metric by 160% and 145% for SHA-1 and SHA-256, respectively.

These methodologies exploit specific properties of hash functions and incorporate optimization techniques such as loop un-rolling, spatial pre-computation and resource re-ordering, retiming, temporal pre-computation and circuit-level optimization (including the use of Carry Save Adders - CSAs).

In the optimized designs of the transformation round, which are shown in Figure 5 and Figure 6, due to the applied balanced loop-unrolling technique, two iterations are unrolled and merged in a new mega transformation round. In the mega transformation round, two iterations are performed in a single clock cycle, so the final hash values are derived after half the number of operations (and half the number of clock cycles), compared to the non-optimized designs. Concerning the transformation rounds implementation (Figure 5 and Figure 6), due to the applied spatial and temporal pre-computation techniques,[19,20] it is composed by the pre- and post-computation units. The purpose of pre-computation unit is to compute several clock cycles before intermediate values which are required by the next iterations of the transformation improving in that way the critical path and throughput. All the above result to much higher throughput designs that improve also the throughput/area cost factor and set the current state-of-the-art in the field.

The general pipeline architecture of the optimized designs is shown in Figure 7. It is similar to that of Figure 6with some variations. It includes an extra unit called *initialization unit*, required by the applied optimization techniques. Its purpose is to provide the required initial values which are needed due to the applied loop-unrolling, retiming, and temporal and spatial pre-computation techniques. Also, due to the loop-unrolling by two operations, each W_{unit} computes two W values per clock cycle. Therefore each W_{unit} includes two identical modules, a 16×32-bit shift register, and a 4-to-2 multiplexer. Also, as the values $W_{t+4} + K_{t+4}$ and $W_{t+5} + K_{t+5}$ are used in the mega block of SHA-1 (Figure 5) extra adders and registers are used to feed these values to each Round unit. The same also holds for the optimized SHA-256 block (Figure 6), where the values $W_{t+3} + K_{t+3}$ and

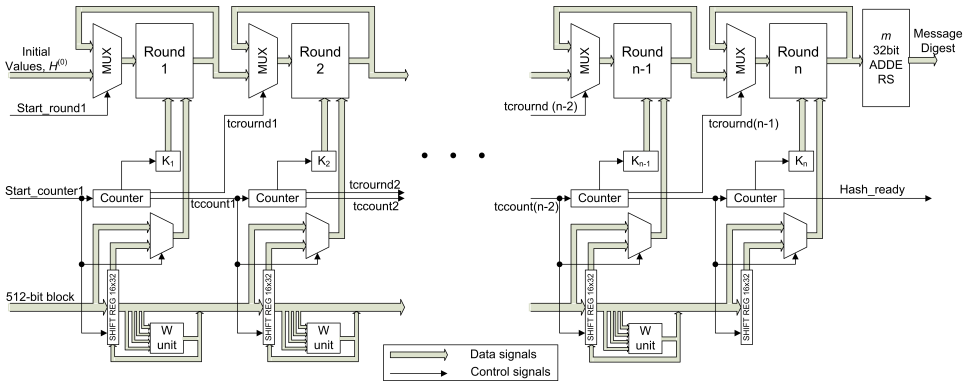

Fig. 4. General pipeline architecture for base SHA-1 and SHA-256 hash functions

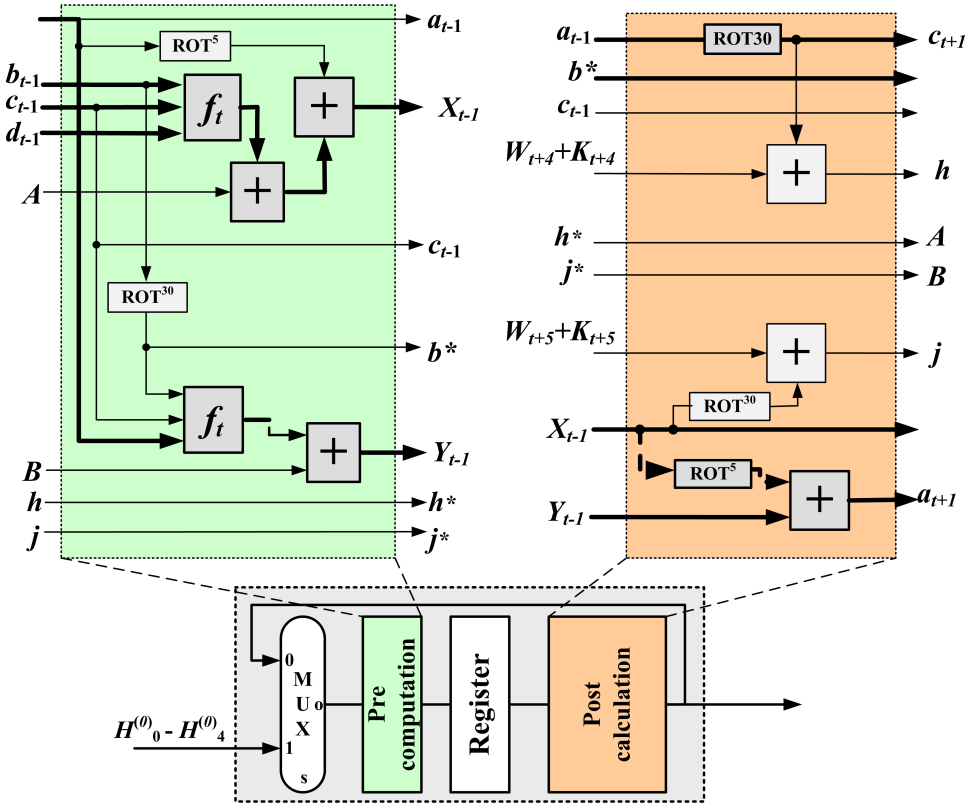

Fig. 5. SHA-1 hash function - Optimized transformation round

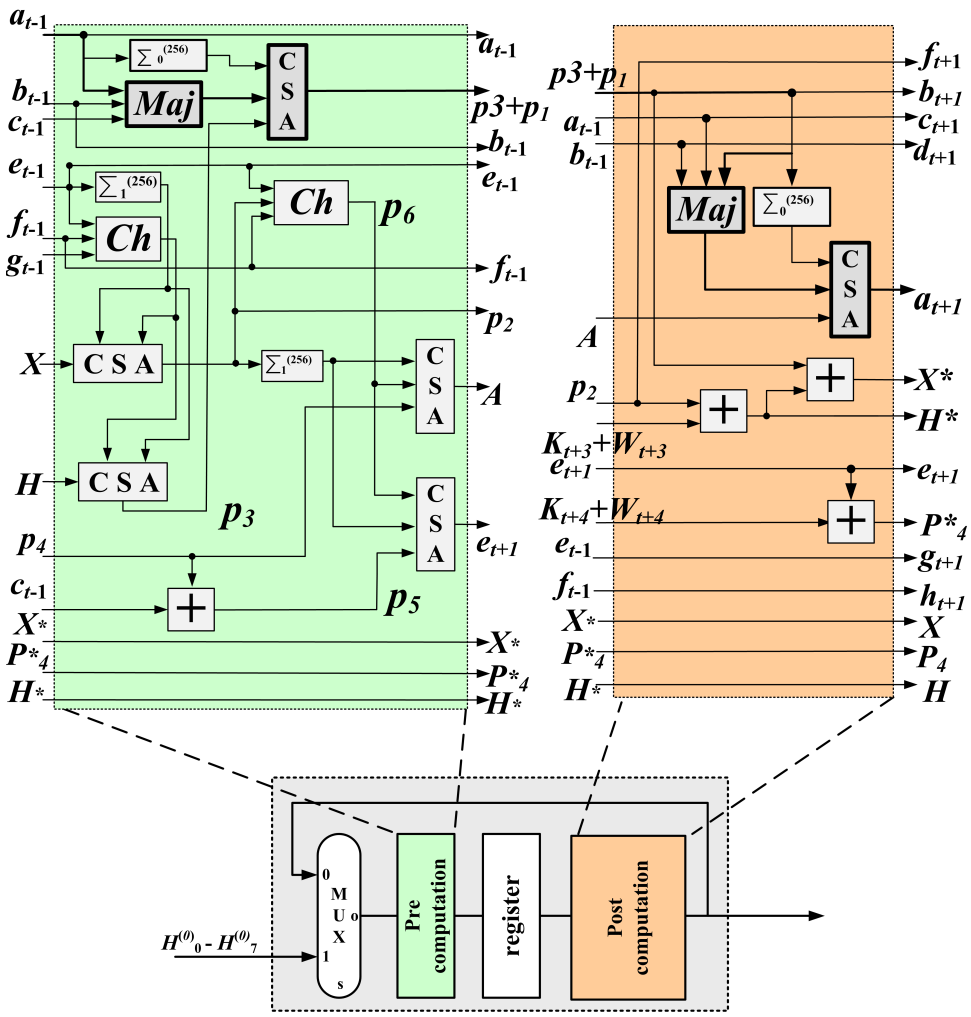

Fig. 6. SHA-256 hash function - Optimized transformation round

$W_{t+4} + K_{t+4}$ are used.

Compared to the base designs, the critical paths of SHA-1 and SHA-256 optimized transformation rounds are shorter as indicated by the darker blocks in Figure 5 and Figure 6 The critical path of the SHA-1 optimized architecture includes two adders and one non-linear function (two similar paths exist, as shown in Figure 5) and a multiplexer, whereas the critical path of SHA-256 includes two non-linear functions, two CSAs, and a multiplexer. When four pipeline stages are used, the optimized SHA-1 core is able to compute a message block's hash value in 40 cycles,[19] while the SHA-256 one in 32 cycles.[20]

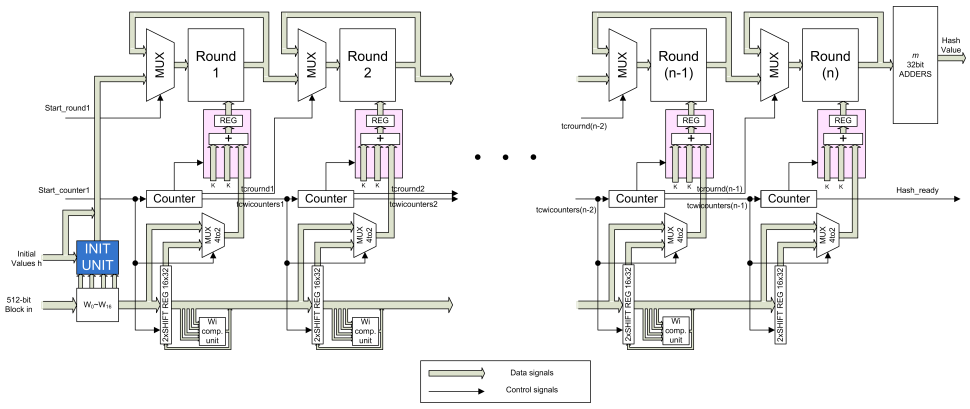

Fig. 7. General pipeline architecture for optimized SHA-1 and SHA-256 functions

5. Experimental Results

The base and optimized architectures for the SHA-1 and SHA-256 functions were captured in VHDL. A set of different designs with varying number of applied pipeline stages was implemented in four Xilinx families, namely the Virtex (xcv1000-6FG680 device), Virtex E (xcv3200e-8FG1156 device), Virtex II (xc2v6000-6FF1517 device for SHA-1 and xc2v6000-5FF1517 device for SHA-256), and Virtex 4 (xc4vlx100-12FF1148 device). These devices were chosen in order to investigate the trade-offs of the SHA-1 and SHA-256 pipelined architectures in a wide range of FPGA technologies, thus offering a more complete study.

Table 1 summarizes the key characteristics for each device. We note that this summary should not be used as a basis for direct comparison of the implemented designs on different families. Such as comparison is misleading and not fair because the FPGAs are quite different: one family is targeting low-power implementations, another family has more complex LUTs, and yet another one has even CLBs with multiple LUTs.

Table 1. FPGA characteristics

Type	System gates	CLB array	Cells and slices
Virtex (xcv1000-6FG680 device)	1,124,022	64×96	27,648 cells
Virtex-E (xcv3200e-8FG1156 device)	4,074,387	102×156	73,008 cells
Virtex II (xc2v6000-6FF1517 device)	6 million	96×88	33,792 slices
Virtex-4 (xc4vlx100-12FF1148 device)	n.a.	192×64	110,592 cells

Note: Information based on the datasheets available at http://www.xilinx.com/support/documentation/data_sheets/{ds003,ds022,ds031,ds112.pdf}.

The XST synthesis tool of Xilinx ISE Design Suite (version 13.1) was used for

mapping the designs to the FPGAs devices. The functionality of the implementations was initially verified via Post-Place and Route simulations using the Model Technologys ModelSim simulator. A large set of test vectors apart from those provided by the standard were used. Also, downloading to development boards and additional functional verification were performed via the ChipScope tool of Xilinx.

The number of the applied pipeline stages in each hash function's design that has to be evaluated, depends on the number of the iterations that this hash function performs. If the number of iterations is divisible by the number of the applied pipeline stages then all the pipeline stages will be fully exploited without need for inserting pipeline stalls (where some pipeline stages do not process any values). Thus, only these numbers of the applied pipelined stages must be evaluated through development of the corresponding hash functions' pipelined designs so as to define the optimum version in each case. In all other versions of pipelined designs there will be, in certain time instances, idle pipeline stages which results to severe degradation of the hash functions design performance.

Using this fact as a design guide, for the SHA-1 base design, ten pipeline versions were developed. These versions correspond to the ones using 1, 2, 4, 5, 8, 10, 16, 20, 40, and 80 pipeline stages, called as SHA1base-p1, SHA1base-p2, SHA1base-p4, etc. Similarly, for the SHA-256 base hash function, 1, 2, 4, 8, 16, 32, and 64 pipeline stages were developed, which are called as SHA256base-p1, SHA256base-p2, SHA256base-p4 etc.

In the optimized architectures, due to loop-unrolling, each pipeline stage performs two iterations in one clock cycle. This way, the hash value for SHA-1 is produced after 40 iterations. Following a similar approach eight versions with 1, 2, 4, 5, 8, 10, 20 and 40 pipeline stages were developed for the SHA-1 optimized hash function, which are called as SHA1opt-p1, SHA1opt-p2, SHA1opt-p4, etc. Similarly, for the SHA-256 optimized architecture, six versions with 1, 2, 4, 8, 16, and 32 pipeline stages were developed and evaluated which are called as SHA256opt-p1, SHA256opt-p2, SHA256opt-p4, etc.

In some pipelined versions of SHA-1, more than one non-linear functions must be incorporated in the round unit. This is because each one of the four non-linear functions of SHA-1 is used only for 20 iterations. In certain cases of pipelined designs each transformation round performs a number of iterations that does not divide 20 completely (without remainder), leading to the need for more non-linear functions in each round unit. For instance, at the four-stage pipeline version each stage (round unit) performs exactly 20 iterations, thus incorporating only one non-linear function. However, if a five-stage design is adopted, then each stage executes 16 iterations. Hence, except the first round unit, the remaining ones contain two non-linear functions In this case, extra multiplexers are inserted in the round unit to choose the correct output among all included non-linear functions. Moreover, a more complex control unit is needed. These SHA-1 pipelined versions consume extra area leading to degradation of throughput/area factor. This does not happen for the SHA-256 function as it uses the same non-linear function in all iterations.

Pipeline Stages	V: Virtex (-6), V-E: Virtex-E (-8), V-II: Virtex-II (-5), V-4: Virtex-4 (-12)											
	Freq.(MHz)				Slices (x10^3)				Throughput (Gbps)			
	V	V-E	V-II	V-4	V	V-E	V-II	V-4	V	V-E	V-II	V-4
1	50.2	58.1	66.46	124.3	0.8 (7%)	0.79 (2%)	0.75 (2%)	0.74 (2%)	0.32	0.37	0.43	0.80
2	44.8	55.2	64.2	109.5	1.18 (10%)	1.21 (4%)	1.12 (2%)	1.11 (2%)	0.57	0.71	0.82	1.40
4	45.9	57.0	71.1	120.7	2.07 (17%)	2.14 (7%)	2.00 (4%)	1.99 (4%)	1.18	1.46	1.82	3.09
5	44.8	55.2	64.1	109.5	2.68 (22%)	2.74 (8%)	2.59 (6%)	2.56 (5%)	1.43	1.77	2.05	3.50
8	45.9	57.3	71.2	120.9	3.92 (32%)	4.03 (12%)	3.82 (8%)	3.93 (8%)	2.35	2.93	3.65	6.19
10	44.6	55.2	64.3	109.6	4.95 (40%)	5.06 (16%)	4.8 (10%)	4.75 (10%)	2.85	3.53	4.12	7.01
16	45.9	57.0	71.1	120.3	7.94 (65%)	8.11 (25%)	7.69 (17%)	7.86 (16%)	4.70	5.84	7.28	12.32
20	44.8	52.8	71.0	110.8	9.57 (78%)	9.43 (29%)	9.86 (21%)	9.00 (18%)	5.73	6.76	9.09	14.18
40		51.1	70.0	110.3		17.69 (55%)	18.65 (40%)	17.56 36(%)		13.08	17.92	28.24
80			71.8	129.6			33.77 (72%)	31.70 (64%)			36.76	66.36

Fig. 8. SHA-1 Base Architecture

The above pipelined designs were implemented in the previously mentioned FPGA technologies and measured in terms of frequency, throughput, area, and throughput/area cost factor. The throughput is calculated as

$$throughput = \frac{\#bits \times f}{\#cycles} \tag{3}$$

where $\#bits$ refer to the number of the processed bits, $\#cycles$ corresponds to the required clock cycles between successive messages to generate each message digest, and f is the operating frequency of the design.

In Figures 8, 9, 10, 11, the experimental results in terms of frequency, area (FPGA slices), and throughput for all the considered pipelined designs (base and optimized) of SHA-1 and SHA-256 functions for four different FPGA devices are presented. It must be stressed that these values are obtained after downloading the designs to the development boards. Also, the speed grade of the used devices (which significantly affects the achieved operating frequency) during synthesis is provided in parenthesis for each FPGA family. Moreover, the Optimization Effort (`opt_level`) constraint of the Xilinx ISE synthesis tools was set to *High* and the Optimization Goal (`opt_mode`) was set to *Speed*. Experimental results were also performed with Optimization Goal (`opt_mode`) set to *Area*. The obtained results showed minor improvements of consumed area but significant reduction of achieved frequency. Overall the results for throughput/area factor were always better when using speed as the optimization goal. For this reason all experimental results were obtained using speed as the optimization goal.

For the cases that the design does not fit in the FPGA device, the corresponding cell in the above Figures appears empty. Also, along with the area, the device utilization is provided in parenthesis.

6. Trade-offs Study

In this section the obtained experimental results of the implemented pipelined versions of SHA-1 and SHA-256 hash functions are analyzed in terms of throughput, area, and throughput/area factors along with the trade-offs obtained for the various

Pipeline Stages	V: Virtex (-6), V-E: Virtex-E (-8), V-II: Virtex-II (-6), V-4: Virtex-4 (-12)											
	Freq.(MHz)				Slices (x10^3)				Throughput (Gbps)			
	V	V-E	V-II	V-4	V	V-E	V-II	V-4	V	V-E	V-II	V-4
1	51.1	53.3	63.8	110.6	0.90 (7%)	0.86 (3%)	0.87 (3%)	0.92 (2%)	0.41	0.43	0.51	0.88
2	46.9	51.8	62.4	108.7	1.33 (11%)	1.36 (4%)	1.44 (4%)	1.58 (3%)	0.75	0.83	1.00	1.74
4	46.9	51.8	62.4	108.8	2.39 (19%)	2.40 (7%)	2.60 (8%)	2.97 (6%)	1.50	1.66	2.00	3.48
8	45.4	50.8	62.1	108.3	4.58 (37%)	4.79 (15%)	5.29 (16%)	6.29 (13%)	2.91	3.25	3.97	6.93
16	44.5	49.6	61.9	108.0	8.99 (73%)	9.09 (28%)	10.66 (32%)	12.12 (25%)	5.70	6.35	7.92	13.82
32		50.1	62.0	108.0		17.81 (55%)	18.98 (56%)	23.52 (48%)		12.83	15.87	27.65
64		58.3	76.3	111.2		29.99 (92%)	27.61 (82%)	31.42 (64%)		29.85	39.07	56.93

Fig. 9. SHA-256 Base Architecture

Pipeline Stages	V: Virtex (-6), V-E: Virtex-E (-8), V-II: Virtex-II (-5), V-4: Virtex-4 (-12)											
	Freq.(MHz)				Slices (x10^3)				Throughput (Gbps)			
	V	V-E	V-II	V-4	V	V-E	V-II	V-4	V	V-E	V-II	V-4
1	65.1	73.6	86.4	161.6	0.96 (8%)	0.99 (3%)	0.9 (2%)	0.89 (2%)	0.83	0.94	1.11	2.07
2	58.3	72.1	83.5	142.3	1.42 (12%)	1.45 (4%)	1.35 (3%)	1.33 (3%)	1.49	1.85	2.14	3.64
4	59.5	74.1	92.4	157.2	2.49 (20%)	2.56 (8%)	2.41 (5%)	2.39 (5%)	3.05	3.79	4.73	8.05
5	58.3	72.1	83.5	142.3	3.22 (26%)	3.28 (10%)	3.10 (7%)	3.07 (6%)	3.73	4.61	5.34	9.11
8	59.5	74.1	92.4	157.2	4.75 (39%)	4.84 (15%)	4.59 (10%)	4.56 (9%)	6.09	7.59	9.46	16.10
10	58.3	72.1	83.5	142.3	5.94 (48%)	6.07 (19%)	5.77 (12%)	5.74 (12%)	7.46	9.23	10.69	18.21
20	56.9	70.1	92.3	144.1	11.06 (90%)	11.31 (35%)	11.86 (25%)	10.83 (22%)	14.57	17.95	23.63	36.89
40		74.9	95.5	172.6		20.17 (62%)	22.00 (47%)	20.19 (41%)		38.35	48.90	88.37

Fig. 10. SHA-1 Optimized Architecture

Pipeline Stages	V: Virtex (-6), V-E: Virtex-E (-8), V-II: Virtex-II (-6), V-4: Virtex-4 (-12)											
	Freq.(MHz)				Slices (x10^3)				Throughput (Gbps)			
	V	V-E	V-II	V-4	V	V-E	V-II	V-4	V	V-E	V-II	V-4
1	61.1	63.2	76.7	132.2	1.18 (10%)	1.10 (3%)	1.16 (3%)	1.22 (2%)	0.98	1.01	1.23	2.12
2	57.3	62.0	74.5	130.1	1.72 (14%)	1.77 (5%)	1.94 (6%)	2.06 (4%)	1.83	1.98	2.38	4.16
4	57.3	62.0	74.5	130.1	3.10 (25%)	3.16 (10%)	3.50 (10%)	3.96 (8%)	3.67	3.97	4.77	8.33
8	54.3	61.0	74.3	130.0	5.77 (47%)	5.89 (18%)	6.62 (20%)	7.78 (16%)	6.95	7.81	9.51	16.64
16	53.4	58.5	74.1	129.7	11.90 (97%)	12.16 (37%)	13.78 (41%)	15.26 (31%)	13.67	14.98	18.97	33.20
32		63.2	79.1	135.3		19.41 (60%)	20.96 (62%)	26.34 (54%)		32.36	40.50	69.27

Fig. 11. SHA-256 Optimized Architecture

numbers of pipeline stages. The throughput/area metric is the fairer factor as it relates the achieved throughput with the consumed area.

As mentioned in Section 4, the critical path of the architecture lies inside the round unit and it should be, somewhat, constant regardless of the number of the pipeline stages. This happens because the rounds units are identical (except for the cases in SHA-1 where more than one non-linear function is integrated in the round) and the architecture is modular. Specifically, the architecture consists of tiles that include the round and W unit, the local registers (memory), and the counter, which are repeated in the horizontal direction. However, the result Figures reveal a variation of the achieved frequency, caused by the different routing delays of each implementation. Indeed, this becomes more pronounced in the Virtex device whose interconnection network is not as rich as the other families. Also, in SHA-1 designs frequency varies more when the round unit includes more than one non-

linear function and the required multiplexer, as discussed earlier.

For all the considered FPGA families, throughput increases almost linearly with the number of pipeline stages. This is justified considering Eq. (3), where the dominant factor is the number of clock cycles, which decreases linearly as the number of pipeline stages increases, whereas the contribution of the frequency variations in changing drastically the linearity is small. Also, both in SHA-1 and SHA-256 implementations, the frequency and throughput are improved when moving to modern FPGA families.

Concerning the occupied area there is no linear relation with the number of pipeline stages due to the special nature of the FPGA devices. As it is known, each FPGA slice contains one or more LUT, multiplexers, and flip-flops to implement logic. Thus, as the number of the pipeline stages increases, the unused resources of the already employed slices can be also used to implement the extra logic, which results in a non-linear increase of area. The tendency it that as the pipeline stages increase, a better area resources utilization is achieved leading to corresponding increased throughput/area factor.

6.1. *Trade-offs of Base Designs*

In Figure 12, the throughput/area graphs for the base SHA-1 and SHA-256 implementations are provided for the number of pipeline stages discussed in Section 5. In addition, in Figure 13 the percentage improvements of throughput/area factor over the non-pipelined (one-stage) implementation are illustrated. The improvement I ratio is computed based on Eq. (4), where T is the througput, A the area, and k is the number of pipeline stages.

$$I = \frac{T_k/A_k}{T_1/A_1} \tag{4}$$

As shown in Figure 12, for SHA-1 base pipelined designs, throughput/area increases for SHA1base-p4, SHA1base-p8, SHA1base-p40, and SHA1base-p80 designs in all devices. Moreover, as shown in Figure 13, the throughput/area ratio does not improve linearly although the whole architecture is modular. Specifically, the throughput/area improvement over the non-pipelined design is reduced for the SHA1base-p5 and SHA1base-p10 designs, compared to all other designs. This is explained by the fact that in these cases the transformation round (Round unit in Figure 4) contains more than one non-linear function and extra multiplexers, which results to an area increase, frequency decrease and correspondingly to reduced throughput/area values.

For all FPGA families, regarding SHA-1, the design with eight pipeline stages achieves higher throughput/area values than the one with 4 pipeline stages, as shown in Figure 13. Specifically, the throughput/area improvements of the eight-staged pipelined designs vs. the four-staged ones over the non-pipelined designs is about 8%, with the eight-stage pipeline design being the best. The design with

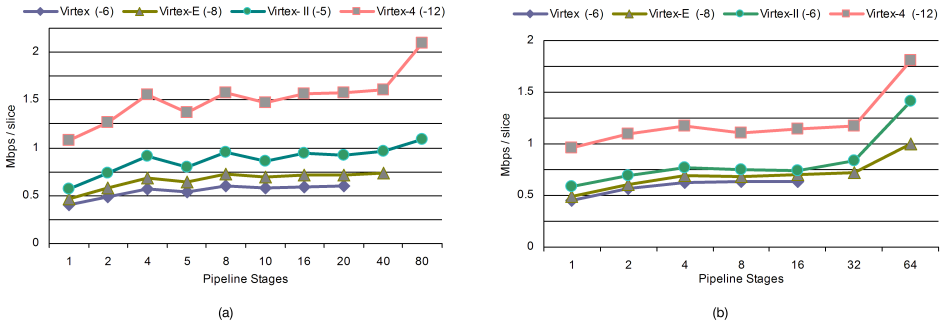

Fig. 12. Base architectures Throughput /Area: SHA-1 (a), SHA-256 (b)

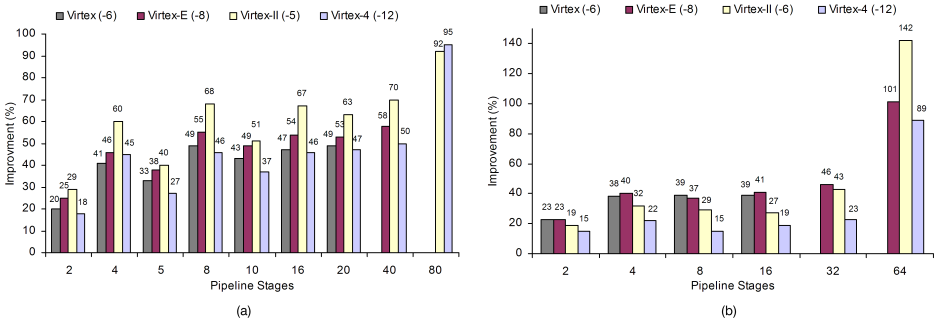

Fig. 13. Base architectures - Throughput/Area improvements over 1-stage pipeline design: SHA-1 (a), SHA-256 (b)

eight pipeline stages is also area-efficient (see Table 8), especially in newer and more spacious FPGA devices like Virtex-4, as it occupies 3,930 of 49,152 slices (8% area utilization). The 16- and 20-stage pipelined designs also have increased throughput/area improvement and may be considered as area efficient in certain cases.

This leads to a first conclusion: more pipeline stages can be utilized to achieve not only higher throughput, which is required by the security protocols integrated in modern communication protocols like IPv6, but also achieve higher throughput/area values and thus better exploitation of the utilized area, without violating the area constraint. This is important as it turns over a critical design choice that has been widely adopted in literature, where the four-stage pipeline dominates all the published implementations for the SHA-1 hash function.

Moreover, the possibility of applying more than eight pipeline stages (e.g., 16 or 20 stages) should be carefully considered, as it greatly increases throughput, while it also achieves improved throughput/area cost factor. Considering the rapid evolution of the FPGA technology, the adoption of this design choice (if the whole system

will fit in the selected device) cannot be rejected nowadays in certain applications.

However, as it can be seen in the Figures with the analytical results, a fully pipelined design with 80 pipeline stages is unrealistic, as it cannot even fit in older and smaller families (like Virtex), while in newer and more spacious but also more expensive devices, (like those in Virtex 4 family), it takes up a significant portion of the whole FPGA device (utilization 31,420 of 49,152 slices, almost 64%). Also, the 80-stage pipeline design achieves a high throughput that cannot be exploited by realistic commonplace applications and thus it is not worth paying the corresponding area penalty.

Comparisons of various SHA-256 base pipelined designs are a little different. The plot lines of Figure 12(b) and Figure 13(b) clearly indicate that a better ratio of throughput/area ratio is achieved when using more pipeline stages. Due to the fact that the same non-linear function is used in all 64 iterations of the algorithm, the plot lines of Figure 12(b) are smoother than those of SHA-1 base graph and do not present extremes. This conclusion is independent of the adopted FPGA family.

In antithesis to the SHA-1 base hash function design, in SHA-256 base design, it is the application of the four pipeline stages that results to higher throughput/area ratio. This highlights the fact that this design decision is not straightforward, but also depends on each certain implemented hash function and its design architecture. Thus, this conclusion verifies the correctness of the decision of four-stage pipeline implementations that has been widely adopted in the designs presented in literature, which in case of SHA-256 has no clear motivation.

However, considering the evaluation results of Figure 12(b) and Figure 13(b), it seems that designers can adopt more pipeline stages for their SHA-256 implementations, thus achieving much higher throughputs in cases that a certain application calls for that extra throughput (e.g. IPSec). Nevertheless, in that case, they will end up with a worse exploitation of the utilized area. In any case the designers can use the performed analysis so as estimate the cost of each pipelined version and based on the area penalty they are willing to pay they can determine how many pipeline stages they will utilize in their SHA-256 implementation.

As it can be seen from the Table 9, the application of a fully pipelined design with 64 pipeline stages is unrealistic since in older and smaller devices like Virtex cannot even fit in the device, while in modern and larger but also more expensive devices like those in Virtex-4 family takes up a very significant portion of the whole area. As in SHA-1, the fully pipelined design achieves a very high throughput that however, cannot be exploited by realistic nowadays applications and thus, it is not worth paying this area penalty. It must be stressed that for both SHA-1 and SHA-256 functions, the increased performances of the fully pipelined designs occur due to the lack of multiplexers between the rounds, resulting to shorter critical paths and thus higher operating frequencies.

In general, throughput and throughput/area improve significantly for all pipeline versions when newer and bigger FPGA families are used, as expected.

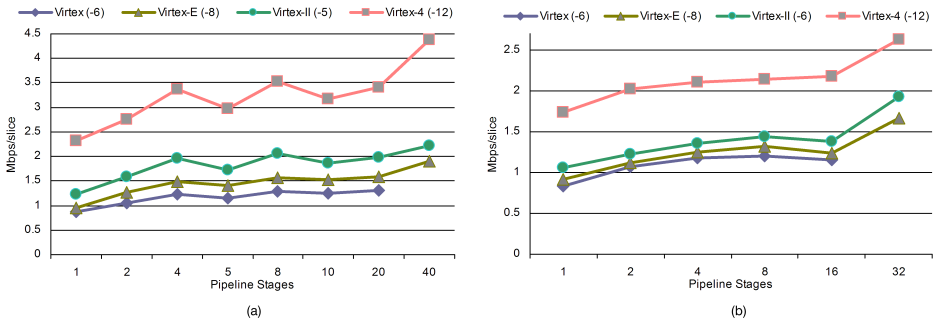

Fig. 14. Optimized architectures - Throughput /Area: SHA-1 (a), SHA-256 (b)

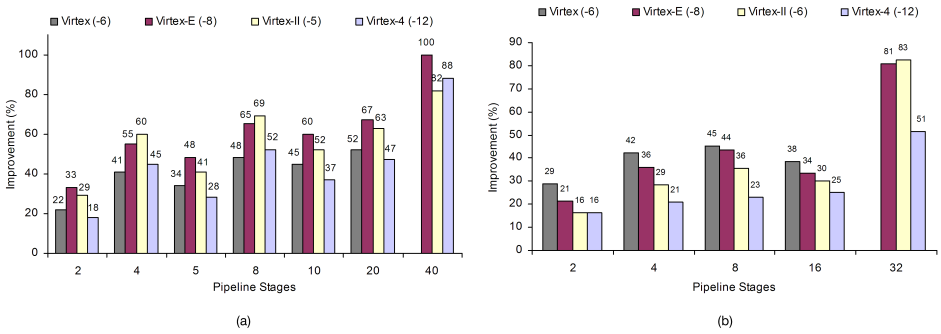

Fig. 15. Optimized architectures - Throughput /Area improvements over 1-stage pipeline design: SHA-1 (a), SHA-256 (b)

6.2. *Trade-offs of Optimized Designs*

Concerning the achieved frequency, throughput and occupied area the conclusions derived for the base designs also hold for the optimized designs of SHA-1 and SHA-256 hash functions.

Due to the routing delay, the achieved frequency exhibits variations among the applied pipeline stages; however, the throughput still increases almost linearly by a factor equal to the number of the utilized pipeline stages. Also, for the reasons explained in Section 6, the area increases non-linearly with the number of the pipeline stages.

The performance trades-offs in terms of throughput/area cost factor of the various pipeline versions of SHA-1 and SHA-256 optimized architectures are shown in Figure 14 and Figure 15. The same conventions as in Figure 12 and Figure 13 are used to present the evaluation results and performance trade-offs.

Comparing the SHA-1 optimized designs, as shown in Figure 14 and Figure 15, an increased performance in terms of throughput/area occurs for the SHA1opt-p4,

SHA1opt-p8, SHA1opt-p20 and SHA1opt-p40 in all FPGA families. It is clear that also in the optimized SHA-1 design the application of 8 pipeline stages results to higher throughput/area ratio compared to the application of four pipeline stages. Similarly to the base designs, the difference between SHA1opt-p4 and SHA1opt-p8 designs in terms of throughput/area improvement over the non-pipeline implementation is 8%, with SHA1opt-p8 being the best. Moreover, eight-stage pipeline design is considered as area efficient (see Table 10) especially in modern FPGA families like Virtex 4.

This leads to the same conclusions as earlier about the general design guide towards adoption of more pipeline stages to hash functions' designs either in their conventional or optimized designs, leading to designs with increased throughput but also with increased throughput/area factor. Also in this case, the design choice of applying four pipeline stages, which has been widely adopted in literature, is turned over.

As expected, the fully pipelined SHA-1 optimized design with 40 pipeline stages dominates all optimized pipelined versions in all devices both in throughput and throughput/area metrics. However, as it can be seen from Table 10, the application of a fully pipelined design is unrealistic for exactly the same reasons that were previously stated in Section 6 for the fully pipelined base SHA-1 and SHA-256 hash functions' designs.

Comparisons among the various SHA-256 optimized pipelined designs are shown in Figure 14(b) and Figure 15(b). The plot lines clearly indicate that also in this case a better ratio of throughput/area is achieved when adopting more pipeline stages.

This conclusion arises no matter what the implementation FPGA family is. Moreover, because of the fact that the same non-linear function is used in all operations, the evaluation line once again presents no extrema. As in SHA-1 optimized design and in antithesis with SHA-256 base design, the application of eight pipeline stages is the best solution since it results to higher throughput/area ratio compared to the application of 4 pipeline stages.

Taking in considerations the experimental results in Figure 15, it is clear that for the same reasons stated previously designers should adopt more pipeline stages to their SHA-256 designs achieving much higher throughputs and area exploitation. This conclusion is very important since it is validated in three out of four cases that were evaluated throughout this work. Hence, not only does it turns over a design decision that has been widely adopted in literature, but also leads to the necessity for performing a separate analysis for the pipeline stages that should be used in high performance designs.

The possibility of applying more than eight pipeline stages (i.e., 16) should also be carefully considered in this case, since it will greatly increase the throughput. However, it will significantly increase the occupied area, leading (in most of the cases) to overall decrease of the resulted throughput/area cost factor. However, as it can be seen from the Table 11, the application of a fully pipelined design with 32

pipeline stages is unrealistic since in older and smaller families like Virtex cannot even fit in the device, while in newer ones like Virtex 4 takes up a significant portion of the whole FPGA device. As in SHA-1 designs, the fully pipelined design achieves a very high throughput that however, cannot be exploited by realistic everyday applications and thus, it is not worth paying this area penalty.

6.3. *Base vs. Optimized Designs*

As it mentioned in Section 2, another question that arises is whether it is worthy to spent effort and area to produce an optimized design or it should be better to use the extra for applying more pipeline stages. In other words, in the base architectures can we achieve the same throughput as in optimized architectures by only increasing the pipeline stages and what happens with occupied area? The optimized SHA-1 and SHA-256 designs have been proposed, evaluated and shown to have significant improvements over base designs when comparing their corresponding four pipeline stages designs in [19] and [20]. However someone could claim that when adopting more pipeline stages, the complexity of the optimized designs might result to decreased improvement. This way there could be certain cases where base designs could perform better than the optimized ones for the same occupied area by exploiting the extra area for more pipeline stages and not for optimizations.

In Figure 16 and Figure 17 the area and throughput graphs for the base and optimized implementations of SHA-1 and SHA-256 hash functions in Virtex 4 and Virtex II FPGAs are illustrated. Considering Figure 16(a), it is clear that the optimized architecture is more efficient in terms of throughput and occupied area than the base one, in all cases, regardless of the number of applied pipeline stages.

For instance, consider the SHA-1 five-stage base design. In this case, the achieved throughput equals to 3.5 Gbps and requires about 2,600 slices. The same throughput can also be achieved by the 2-stage optimized design. However, in this case the consumed area equals to 1,300 slices which corresponds to 50% less occupied area compared to the five-stage base design. Moreover, even increasing the pipeline stages of the base architecture, the area penalty remains almost double. Specifically, for the ten-stage base implementation, the achieved throughput equals to 7 Gbps consuming 4,800 slices, whereas the 4-stage optimized design achieves higher throughput (8.1 Gbps) with the half area (2,400 slices).

Similar conclusions are derived for the implementations on the older Virtex II FPGA family. A throughput equal to 2.1 Gbps can be achieved by both five-stage base and two-stage optimized designs. However, in the former case 2,600 slices are required, whereas in the second the area penalty is 1,400 slices.

Concerning the SHA-256 function designs similar observations are derived studying Figure 17. For instance, the eight-stage base implementation achieves a throughput equal to 6.9 Gbps spending 6,300 slices, whereas the four-stage pipelined optimized design achieves an 8.3 Gbps throughput and consumes about 4,000 slices.

Based on the above, it is clear that also the SHA-256 optimized implementa-

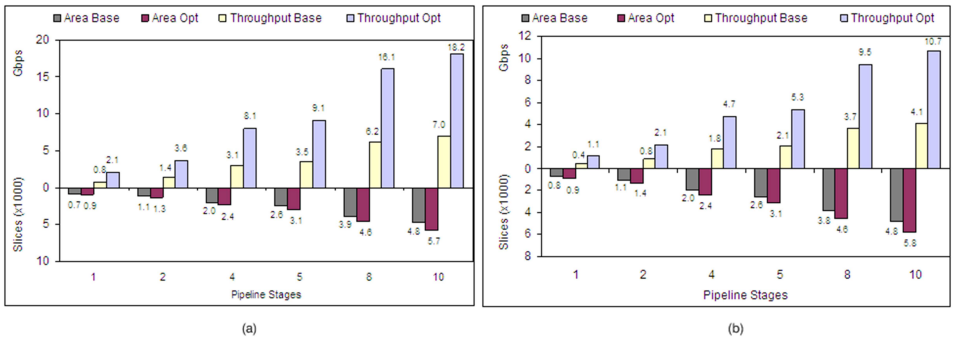

Fig. 16. Area, throughput of SHA-1 hash function implementations: Virtex-4 (a), Virtex-II (b)

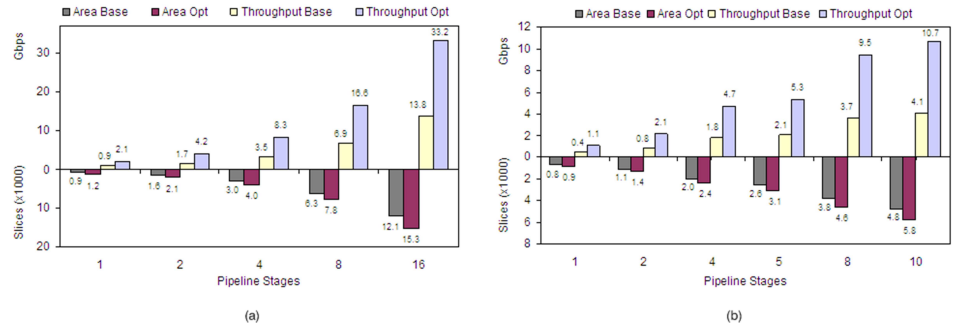

Fig. 17. Area, throughput of SHA-256 hash function implementations: Virtex-4 (a), Virtex-II (b))

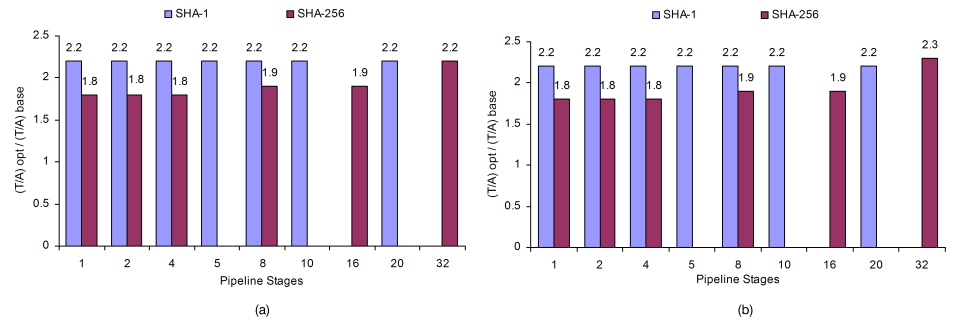

Fig. 18. Throughput/area (T/A) ratio of optimized and base SHA-1 and SHA-2 hash function implementations: Virtex,4 (a), Virtex,II (b)

tions always outperform the base ones when both throughput and area taken into account. For clarity reasons a reduced number of pipeline versions are depicted in Figure 16 and Figure 17. A more global picture is provided in Figure 18. In partic-

ular, we provide graphs that represent the throughput/area ratios of the optimized and base implementations for different number of pipeline stages. The vertical axis represent the (throughput/area)_opt / (throughput/area)_base ratio.

It is evident, that the optimized designs outperform the base ones regardless of the number of applied pipeline stages in both Virtex-4 and Virtex-II FPGA families. In particular, for the SHA-1 hash function, the achieved throughput/area of the optimized designs is 2.2 times higher than the base designs in both Virtex-4 and Virtex-II implementations. The same also holds for SHA-256 hash function, where the throughput/area values of the optimized designs are 1.8 or 1.9 times higher than the base ones in Virtex-4 and Virtex-II implementations.

Hence, based on the above analysis it is concluded that even though an extra effort is paid for developing sophisticated designs such as those in [19,20], the benefits are significant as the throughput/area metric is drastically increased compared to the base implementations. It is obvious that in case of applications calling for extra throughput, this should be gained by paying the area penalty for optimized designs and not for applying more pipeline stages to our design, if area is considered as constraint. The choice of applying more pipeline stages for throughput increment should be considered only in the case where extra throughput is needed only after the optimized designs have been deployed in the targeted security schemes.

7. Design Guides and Discussion

In previous sections an analysis of trade-offs in terms of area, throughput, and throughput/area among the various pipelined versions of the base and the optimized pipeline architectures of SHA-1 and SHA-256 hash functions was performed. Several Virtex FPGA families have been used for implementing the various pipelined designs making the above analysis holistic and fair leading to a number of important conclusions.

First, the widely-adopted decision about the four-stage pipeline implementations is turned-over. It has been shown that in most cases the designs with eight pipeline stages outperforms that with the four pipeline stages in term of throughput/area cost factor (and of course in terms of throughput as it is expected). Moreover it has been revealed that the number of applied pipeline stages is a design parameter that should be analyzed through validation in the targeted implementation device. This way better exploitation of used area resources can be achieved but also applications calling for high throughput can be efficiently realized in FPGAs through integration of the optimum number of pipeline stages in each case.

Concerning the optimized designs it was derived that they clearly outperform the base ones in all cases of applied pipelined stages, when both the throughput and throughput/area factor are taken into consideration. This is also an important conclusion as it recompenses the extra effort required to develop an optimized design. Moreover, it occurs that if a certain amount of integration area can be sacrificed for speeding-up a hash function's design, this area should be used for

algorithmic optimizations like those in [19] and [20] rather than for applying more pipeline stages.

Also it was derived that designs with more than eight pipeline stages are also very promising in certain cases, taking into account the evolution of the FPGA technology. Specifically, designers aiming at achieving high throughput FPGA implementations of SHA-1 and SHA-256 hash functions without having hard area constraints could adopt the SHA1opt-p20 and SHA256opt-p16 choices. However fully pipelined designs are unrealistic since in older and smaller families like Virtex cannot even fit in the device, while in newer ones like Virtex-4 take up a significant portion of the whole FPGA device. It must be considered that hash functions, in most cases, are incorporated in a bigger security system like a security protocol (e.g., IPSec). Hence, the hash function design itself has to have reasonable area consumption. Those who are concerned about area but have to achieve an increased performance design are encouraged to exploit SHA1opt-p4 and SHA256opt-p4. In general, four-stage pipelined designs are the most balanced ones concerning the throughput/area cost ratio in conjunction with their area consumption.

The work in this paper, in conjunction with similar works[16,30,31] for AES can be used as a designers' guide for developing efficient designs in terms of performance, throughput, and area of widely used security schemes in form of hardware accelerators for network security schemes, e.g., IPSec for IPv6.

Specifically, IPSecs functionality and bottleneck is determined by the incorporated HMAC mechanism and block cipher algorithm. HMAC mechanism, utilizes two hash cores in its implementations. Thus from Figures 9 and 11, a rough estimation about the consumed area for HMAC (something more than 2 times the area of each optimized hash function), and the achieved throughput (the same as the incorporated optimized hash function, see Eq. (3)) occurs. It has been proved that in any case only the optimized hash functions (that is those reported in Figures 9 and 11) should be considered for integration in high-throughput demanding applications. For example, after a thorough analysis of the AES implementation results in [16,30,31], an IPSec designer can decide (depending on the specific development board at his availability), how many pipeline stages will he adopt for his/her AES and hash implementation, so as to end up with a balanced, high-throughput for IPSec's hardware accelerator.

On the other hand if a designer realizes a different security protocol which incorporates different security mechanisms (i.e., based only on a single hash function and not on HMAC mechanism), then the designer of this protocol will end up with different design choices for optimization of the specific protocol's design.

Finally an issue that should be clarified is that all pipelined designs of hash functions in order to function correctly they must be supplied either directly with only one-block messages or via a scheduler for messages that consist of more blocks.[42] This is because in multi-block messages, the computation of every next block begins from the hash value of the previous block and not from values (which initiate the

computation of the first block only). For processing of multi-block messages with pipelined hash designs, where m pipeline stages have been applied, a scheduler should be used so as to enable full exploitation of all pipeline stages.

In such cases, m different messages can be concurrently processed, but each pipeline stage must process blocks from m different messages at each time instance. The scheduler will arrange consecutive feeding of the hashing core with blocks from different messages and will synchronize the processing of the next block of a message with the process finalization of the previous block of the same message in the hash core. This way all the pipeline stages will always be fully exploited, and thus the reported values of achieved throughputs are valid. The development of the above scheduler, which can be easily implemented in software without affecting the presented results, is outside of the aims and scope of this paper.

We note that there is no reason to implement the message scheduler in hardware. The corresponding software component does not affect the operating frequency or the achieved throughput and does not impose any security issues.[20] To the best of our knowledge, no published work considers a message scheduler as a viable option for implementing hash functions in hardware.

8. Conclusions

In this paper, a detailed study and evaluation of various pipelined -base and optimized- designs of SHA-1 and SHA-256 hash functions in terms of several performance factors was presented. Certain design choices were emerged, while it was also proved that optimized designs always outperform the base ones regardless of the number of applied pipeline stages. The performed analysis, in conjunction with similar works on AES, provides useful design guidelines, towards the implementation of high throughput security schemes like IPSec.

Acknowledgments

The authors would like to thank Dr. Artemios G. Voyiatzis for typesetting support and suggestions that improved presentation clarity.

References

References

1. FIPS 198, the keyed-hash message authentication code (HMAC) federal information processing standard (2002), NIST Publication, US Dept. of Commerce.
2. SP 800-32, introduction to public key technology and the federal PKI infrastructure (2001), NIST Publication, US Dept. of Commerce.
3. L. Loeb, *Secure Electronic Transactions: Introduction and Technical Reference* (Artech House Publishers, 1998).
4. D. Johnston and J. Walker, Overview of IEEE 802.16 security, *IEEE Security & Privacy* **2**(3) (2004) 40–48.

5. FIPS 186-3, the digital signature standard (DSS) federal information processing standard (2009), NIST Publication, US Dept. of Commerce.
6. S. Thomas, *SSL & TLS Essentials: Securing the Web* (John Wiley and Sons Publications, 2000).
7. SP800-77, Guide to IPSec VPNs (2005), NIST Publication, US Dept. of Commerce.
8. P. Loshin, *IPv6: Theory, Protocol and Practice* (Elsevier Publications: USA, 2004).
9. Available pool of unallocated IPv4 internet addresses now completely emptied the future rests with IPv6 ICANN Press Release, (2011).
10. FIPS 197, advanced encryption standard (AES) (2001), NIST Publication, US Dept. of Commerce.
11. Fips 180-3, secure hash standard (shs) (2008), NIST Publication, US Dept. of Commerce.
12. X. Wang, Y. L. Yin and H. Yu, Finding collisions in the full SHA-1, in *Advances in Cryptology–CRYPTO 2005*, (Springer, 2005), pp. 17–36.
13. H. Dobbertin, The status of MD-5 after a recent attack RSAlabs' CryptoBytes, (1996).
14. B. Preneel, Cryptographic hash functions and the SHA-3 competition Talk in AsiaCrypt 2010 Conference, (2010).
15. NIST, Cryptographic hash algorithm competition - SHA-3 (2010).
16. A. Hodjat and I. Verbauwhede, Area-throughput trade-offs for fully pipelined 30 to 70 gbits/s AES processors, *Computers, IEEE Transactions on* **55**(4) (2006) 366–372.
17. H. Michail, Cryptography in the dawn of IPv6 IEEE Goldrush Newsletter, (2010).
18. J. M. Granado-Criado, M. A. Vega-Rodríguez, J. M. Sánchez-Pérez and J. A. Gómez-Pulido, A new methodology to implement the AES algorithm using partial and dynamic reconfiguration, *INTEGRATION, the VLSI journal* **43**(1) (2010) 72–80.
19. H. E. Michail, A. P. Kakarountas, A. S. Milidonis and C. E. Goutis, A top-down design methodology for ultrahigh-performance hashing cores, *Dependable and Secure Computing, IEEE Transactions on* **6**(4) (2009) 255–268.
20. H. E. Michail, G. S. Athanasiou, V. Kelefouras, G. Theodoridis and C. E. Goutis, On the exploitation of a high-throughput SHA-256 FPGA design for HMAC, *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* **5**(1) (2012) p. 2.
21. R. Chaves, G. Kuzmanov, L. Sousa and S. Vassiliadis, Cost-efficient SHA hardware accelerators, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* **16**(8) (2008) 999–1008.
22. Y. K. Lee, H. Chan and I. Verbauwhede, Throughput optimized SHA-1 architecture using unfolding transformation, in *Application-specific Systems, Architectures and Processors, 2006. ASAP'06. International Conference on*, (IEEE, 2006), pp. 354–359.
23. L. Dadda, M. Macchetti and J. Owen, The design of a high speed ASIC unit for the hash function SHA-256 (384, 512), in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, **3**, (IEEE, Paris, France, 2004), pp. 70–75.
24. M. Macchetti and L. Dadda, Quasi-pipelined hash circuits, in *Computer Arithmetic, 2005. ARITH-17 2005. 17th IEEE Symposium on*, (IEEE, 2005), pp. 222–229.
25. N. Sklavos, P. Kitsos, E. Alexopoulos and O. Koufopavlou, Open mobile alliance (OMA) security layer: Architecture, implementation and performance evaluation of the integrity unit, *New Generation Computing* **23**(1) (2005) 77–100.
26. J. Diez, S. Bojanic, C. Carreras and O. Nieto-Taladriz, Hash algorithms for cryptographic protocols: FPGA implementations, in *Proc. TELFOR*, (s.n., Belgrade, Yugoslavia, 2002), pp. 26–28.
27. E.-H. Lee, J.-H. Lee, I.-H. Park and K.-R. Cho, Implementation of high-speed SHA-1

architecture, *IEICE Electronics Express* **6**(16) (2009) 1174–1179.
28. R. P. McEvoy, F. M. Crowe, C. C. Murphy and W. P. Marnane, Optimisation of the SHA-2 family of hash functions on FPGAs, in *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*, (IEEE, 2006), pp. 6–pp.
29. M. Rogawski, X. Xin, E. Homsirikamol, D. Hwang and K. Gaj, Implementing SHA-1 and SHA-2 standards on the eve of SHA-3 competition Talk in 7th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'09), (2009).
30. G. P. Saggese, A. Mazzeo, N. Mazzocca and A. G. Strollo, An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm, in *Field Programmable Logic and Application*, eds. P. Y. K. Cheung and G. A. Constantinides (Springer, 2003) pp. 292–302.
31. J. Zambreno, D. Nguyen and A. Choudhary, Exploring area/delay tradeoffs in an AES FPGA implementation, in *Field Programmable Logic and Application*, eds. J. Becker, M. Platzner and S. Vernalde (Springer, 2004) pp. 575–585.
32. J. Lu and J. Lockwood, IPSec implementation on Xilinx Virtex-II Pro FPGA and its application, in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, (IEEE, 2005), pp. 158b–158b.
33. N. Sklavos and O. Koufopavlou, Implementation of the SHA-2 hash family standard using FPGAs, *The Journal of Supercomputing* **31**(3) (2005) 227–248.
34. E. Khan, M. W. El-Kharashi, F. Gebali and M. Abd-El-Barr, Design and performance analysis of a unified, reconfigurable HMAC-hash unit, *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART 1 REGULAR PAPERS* **54**(12) (2007) p. 2683.
35. M. Kim, J. Ryou and S. Jun, Compact implementation of SHA-1 hash function for mobile trusted module, in *Information Security Applications*, eds. K.-I. Chung, K. Sohn and M. Yung, *Lecture Notes in Computer Science* **5379** (Springer, Berlin Heidelberg, 2009) pp. 292–304.
36. R. Glabb, L. Imbert, G. Jullien, A. Tisserand and N. Veyrat-Charvillon, Multi-mode operator for SHA-2 hash functions, *journal of systems architecture* **53**(2) (2007) 127–138.
37. M. Zeghid, B. Bouallegue, M. Mach-Hout, A. Baganne and R. Tourki, Architectural design features of a programmable high throughput reconfigurable SHA-2 processor, *Journal of Information Assurance and Security* **3**(2) (2008) 147–158.
38. M. Kim, J. Ryou and S. Jun, Efficient hardware architecture of SHA-256 algorithm for trusted mobile computing, in *Information Security and Cryptology*, (Springer, 2009), pp. 240–252.
39. C. Inc., SHA-1 Core, SHA-256 Core, Commercial IP Datasheet (2012).
40. Cadence, Hashing Algorithm Generator SHA-256, Commercial IP Technical Datasheet (2012).
41. H. T. Ltd., Data security products, commercial IPs, datasheets (2012).
42. H. Michail, L. Ioannou and A. Voyiatzis, Pipelined SHA-3 implementations on FPGA: Architecture and performance analysis, in *Proceedings of Second Workshop on Cryptography and Security in Computing Systems (CS2 '15)*, (ACM, 2015), pp. 13:13–13:18.

Ref.: Ms. No. WSPC-JCSC-D-15-00085
AREA-THROUGHPUT TRADE-OFFS FOR SHA-1 AND SHA-256 HASH FUNCTIONS' PIPELINED DESIGNS
Journal of Circuits, Systems, and Computers

Dear Dr. MICHAIL,

Reviewers have now commented on your paper. You will see that they are advising that you revise your manuscript. If you are prepared to undertake the work required, I would be pleased to reconsider my decision.

For your guidance, reviewers' comments are appended below.

If you decide to revise the work, please submit a list of changes or a rebuttal against each point which is being raised when you submit the revised manuscript.

Your revision is due by Oct 18, 2015.

To submit a revision, go to http://wspc-jcsc.edmgr.com/ and log in as an Author. From the main menu, select the item "Submissions Needing Revision" and you will find your submission in that folder.

Yours sincerely

Maurizio Martina, Ph.D
Handling Editor
Journal of Circuits, Systems, and Computers

Ref.: Ms. No. WSPC-JCSC-D-15-00085
AREA-THROUGHPUT TRADE-OFFS FOR SHA-1 AND SHA-256 HASH FUNCTIONS' PIPELINED DESIGNS
Journal of Circuits, Systems, and Computers

ANSWERS TO EDITOR'S AND REVIEWERS' COMMENTS

Dear Dr. Maurizio Martina,

We would like to thank the two reviewers and you for providing valuable feedback towards improving the presentation of our submission.

We have completed the revision based on the reviewers' comment and we address in detail the concerns raised in the following paragraphs.

We feel the need to clearly state the sole scope of our work which is to exploit pipelining in the best way that can lead to implementations with the best throughput/area ratio, which is of great importance to the market. However it also provides the design space of potentially achieved throughputs by choosing a different number of applied pipeline stages, so that someone can figure out the performance limits. Moreover, in this case we also inform about the area cost and the deviation from the best solution in each case.

We are looking forward to receiving your final decision.

Thank you,
Dr. Harris E. Michail

Reviewers' comments:

Reviewer #2: The paper is well written. The topic under discussion is well explained. I think the authors have put great efforts to complete it in all aspects now.

> **Thank you for the effort put into studying the paper and providing a thorough review and for appreciating the work we have put in this work!**

Reviewer #3: Two points are still not completely addressed in my opinion.

Replying to one of the reviewer's comment regarding the motivations of the work, authors state they give enough evidences of the need of their research in page 2. There, they basically say IPv6/IPsec and TLS mandate hash functions, commercial routers have gigabit ports, consequently the performance of hash functions have to be improved. While I agree with the two initial statements, I still see no evidence that the hash function is the limiting speed factor in these products. I will appreciate to see at least a reference to one existing system where the issue is considered the limiting factor.

Dear reviewer,
We would like to thank you for your comments and time for reviewing our paper.

Allow us to provide further evidence justifying the motivation of our work pointing out that there is a great range of achieved throughput mentioned in the submitted manuscript, ranging from 0.32 Gbps (SHA-1 base, no pipeline, in Xilinx Virtex) up to 88.37 Gbps (SHA-1 optimized, 40 pipeline stages, in Virtex 4).

Our work provides the exploration space for the designs of the two most widely used hash functions as long as the application of the pipelining technique is concerned. We aim to scientifically document the answer to a question that is quite often posed in this field: What is the best number of pipeline stages to be applied, so that the optimum design will occur in terms of throughput/area ratio?

This is the sole scope of this work, since up to now; no work exists to answer this question and all researchers in the field either neglect this design choice or they are not justifying their choice. Initially, researchers in the field either adopted no-pipeline designs or four pipeline designs biased by the fact that the initial algorithms in the field (MD-5 and SHA-1) used four different operations, thus leading to a four pipeline design as a straight-forward design choice that would also quadruple throughput. However this fact changed in SHA-256 and SHA-512.

Our opinion is that is in not correct and a full analysis on the impact of the different number of applied pipelined stages is necessary to justify this design choice in all designs. This is true even if we are talking for a high throughput design or a low throughput design. We have to know which is the penalty we pay for the deviation for the optimum solution. We believe that is the basic motivation for our work since hardware designs of hash functions are used in many commercial applications and a great number of researchers are conducting research in this filed.

Now, as long as the need for high throughput implementations is concerned. As already said, the achieved high throughput is essential for certain cryptographic applications that lie in the domain of security; IPSec and SSL/TLS and others. But allow us to highlight the situation taking in consideration IPSec for IPv6, the importance of which is greatly acknowledged [7],[8],[9],[10[,[11].

Hash functions are basic components in the corresponding security scheme of IPSEC and SSL/TLS, offering authentication. However in these security schemes encryption modules are also included like AES. As it shown in Figure 1, many AES (encryption module) implementations achieve very high throughputs, in the range of 20 Gbps and are also operating in very high operating frequencies. So, designing high throughput hashing cores (authentication module), of similar range, is something that will increase throughput of the whole security scheme.

Implementation comparison when Block RAM is not used					
Design	Device	Slices	BRAM	Throughput	Mbps per Slice
Elbirt et al [1]	XCV1000-4	10992	-	1.94 Gbits/s	0.17
Standaert et al [4]	XCV3200E-8	15112	-	18.56 Gbits/s	1.2
Jarvinen et al [6]	XC2V2000-5	10750	-	17.8 Gbits/s	1.66
Design with 4 stages in round	XC2VP30-7	12450	-	21.54 Gbits/s	1.7
Design with 7 stages in round	XC2VP20-7	9446	-	21.64 Gbits/s	2.3

[1] A. Elbirt et al, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists", IEEE Transaction of VLSI Systems 9.4, pp. 545-557, August 2001.
[2] Gaj et al, "Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Arrays", CT-RSA 2001, LNCS 2020, pp. 84-99.
[3] McLoone et al, "High Performance Single-Chip FPGA Rijndael Algorithm Implementations", CHES 2001, Paris, France, 2001.
[4] Standaert et al, "Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs", CHES 2003, LNCS 2779, pp. 334-350, 2003.
[5] Saggese et al, "An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm", FPL 2003, LNCS 2778, pp. 292-302, 2003.
[6] Jarvinen et al, "A fully pipelined memory less 17.8 Gbps AES-128 encryptor", International Symposium on Field Programmable Gate Arrays, pp. 207-215. 2003.

* from paper of A. Hodjat, and I.Verbauwhede, "A 21.54 Gbits/s Fully Pipelined AES Processor" on FPGA IEEE Symposium on Field-Programmable Custom Computing Machines Systems, (FCCM '04) pp. 308-309, 2004.

Fig1. Throughput and Area performance of AES implementations

A similar security scheme is also used in Secure Sockets Layer (SSL) and Transport Layer Security (TLS, which is in common use for both virtual private networks (VPNs) and e-commerce applications. SoC designers are therefore finding the need to build high performance SSL record processing engines in gateways, routers, switches and SSL acceleration appliances to support this requirement.

Moreover this issue has been analytically documented in [7], highlighting the need for high throughput designs of hash functions. Furthermore authors of this work have developed a near-commercialization prototype platform which uses cryptographic algorithms for online distribution of e-content, in which the need for high throughput designs of hash functions was also highlighted [12],[13].

[7] H. Michail, Cryptography in the dawn of IPv6 IEEE Goldrush Newsletter, (2010).
[8] CERF VINT. 2010. Vint Cerf pushes for NZ IPv6 transition., ComputerWorld The voice of the ICT community, Portal. Press Room. http://computerworld.co.nz/news.nsf/news/vint-cerf-pushes-for-nz-ipv6-transition
[9] DORASWAMY N. AND HARKINS D. 2003. IPSec – The New Security Standard for the Internet, Intranets and Virtual Private Networks. Second Edition. Prentice-Hall PTR Publications, New Jersey, USA.
[10] PERSET, K. 2008. Internet Address Space: Economic Considerations in the Management of IPv4 and in the Deployment of IPv6. Ministerial Background Report by Organization for Economic Co-Operation and Development. OECD Ministerial Meeting on the Future of the Internet Economy.

[11] POUFFARY Y. 2000 IPv6 Networking for the 21st Century. In Proceedings of All IP Workshop - IPv6 Advantages, 2000.
[12]G. S. Athanasiou, H. E. Michail, A. Gregoriades and M. Ioannides, "Evolution of the e-Museum Concept through Exploitation of Cryptographic Algorithms", in Proc. of Euromed 2012 - International Conference on Cultural Heritage and Digital Libraries, Lemesos, Cyprus, pp. 291-300, Oct. 2012.
[13]H. E. Michail, C. Louca, D. Gavrilis, A. Gregoriades, L. Anastasiou, and M. Ioannides, "Distribution of Cultural Content through Exploitation of Cryptographic Algorithms and Hardware Identification", in Proc. of Euromed 2014 - International Conference on Cultural Heritage and Digital Libraries, Lemesos, Cyprus, pp. 156-165, Nov. 2014.

The following of the paper addresses FPGA implementations without discussing the system integration of the hash functions; The reader has no idea if the proposed solution can be profitable.

The paper has a different scope. Integration issues have been presented by other researchers (author of this work included) in other papers. This paper is already length and including this information would lead to a not robust paper.
In this paper, we felt that we had to fully analyze and highlight only the impact of the number of applied pipelined stages in the design.
For integration issues please refer to our following paper:

1) **H.E. Michail, G. S. Athanasiou, V. Kelefouras, G. Theodoridis and C. E. Goutis, "On the exploitation of a high-throughput SHA-256 FPGA design for HMAC", ACM Transactions on Reconfigurable Technology and Systems (TRETS) , vol. 5, no.1 , pp. 2:1–2:28, 2012.**
2) **H. E. Michail, G. S. Athanasiou, A. A. Gregoriades, Ch. L. Panagiotou, C.E. Goutis, "High Throughput Hardware/Software Co-design Approach for SHA-256 Hashing Cryptographic Module in IPSec/IPv6", Global Journal of Computer Science and Technology , vol.10, iss. 4, pp.54–59, 2010.**

We strongly believe that the reader and the cryptography practitioner can gain very important knowledge of how the proposed solution can be profitable for his/her system. The throughput and area limits along with the best throughput/area ratio are well defined both for SHA-1 and SHA-256. Moreover the trade off of making a selection different from the best one (i.e. because of area limitations) can be derived from the results in various graphs and tables that exist in the paper.

In the same perspective, the answer to the message scheduler issue, is not completely satisfactory: literature analysis can neglect this aspect, but a coprocessor has to be integrated in a system and, especially when data rates are considerable, memory bandwidth and messages latency become dominant factors potentially making architectural improvements pointless.

We would agree with the reviewer if we were proposing a hw/sw co-deign of the whole crypto-processor. However we have to state that at this paper we are not conducting research for a co-processor or anything near that. In fact we are only conducting research for a sole component of this system (hash functions), implemented in

hardware, and only for one optimization parameter (pipelining) aiming to fully draw the optimization options for just this case. Nowhere in this paper do we mention that we are analyzing or referring to coprocessor.

From our side of view, we believe that the software scheduler (we explained in our previous answer why this scheduler must be implemented in software) is part of a paper discussing the implementation of the whole crypto-processor and all of its components either in hardware in software. A software scheduler wouldn't match the rest of this paper and would distract the reader.

We plan to present the software scheduler in another paper of ours along with other components that comprise the whole cryptoprocessor including AES.

The second open issue relates with side channel attacks. At the best of my knowledge any hash function can be used for HMAC (HMAC-SHA1 or HMAC-SHA2 for instance). Several papers discusses differential power analysis and timing side channels attacks to these algorithms. In my opinion, an architectural proposal should be proved to be at least as secure as the basic implementation it is supposed to replace.

You are absolutely right that a hash function is used to the keyed HMAC. However we note that we didn't impose any alternation to the implemented algorithm. The Hash function was designed according to the corresponding FIPS PUB standard, so there is no reason that the implemented in hardware hash functions are less secure than the proposed algorithms imply.

As already stated, the hardware architecture decisions by no means influence the security of the whole system and thus, a security analysis does not deem necessary. Pipelining is a major hardware optimization technique and it has not been proven or suspected to be introducing side-channel vulnerabilities that do not exist in the base design.

However we will seriously consider your point and investigate the possibility of conducting research in the field in the near future.